AUTOGEN:
The Mars 2001 Odyssey and the "Autogen" Process

By Roy Gladden

Abstract

In many deep space and interplanetary missions, it is widely recognized that the scheduling of many commands to operate a spacecraft can follow very regular patterns. In these instances, it is greatly desired to convert the knowledge of how commands are scheduled into algorithms in order to automate the development of command sequences. In doing so, it is possible to dramatically reduce the number of people and work-hours that are required to develop a sequence. The development of the "autogen" process for the Mars 2001 Odyssey spacecraft is one implementation of this concept. It combines robust scheduling algorithms with software that is compatible with pre-existing "uplink" software, and literally reduced the duration of some sequence generation processes from weeks to minutes. This paper outlines the "autogen" tools and processes and describes how they have been implemented for the various phases of the Mars 2001 Odyssey mission.

What is autogen?

The term "autogen" is applied in two different ways. First, "autogen," in its broadest sense, identifies a process that may be used to automatically generate sequences for a spacecraft and, second, it is a Solaris script that has been used to facilitate this process. By using the "autogen" script and process, a user can rapidly build sequences for a spacecraft that may be lengthy and complicated.

The "autogen" process has been specifically developed to facilitate the generation of sequences where spacecraft commands and blocks are scheduled in a repeatable or well-understood fashion. Originally implemented for the Mars 2001 Odyssey spacecraft, the "autogen" process can be used to build sequences for various missions and their mission phases, including interplanetary cruise, aerobraking, and science operations.

Where did it come from?

The "autogen" process and tools are a direct extension of efforts made in support of Jet Propulsion Laboratory (JPL) missions prior to the Mars 2001 Odyssey, including Mars Observer, Mars Global Surveyor (MGS), and the failed Mars '98 missions. In those and other missions, it was recognized that the scheduling of many commands for certain mission phases tended to follow regular patterns. Therefore, rather than requiring many people to spend several weeks manually building the commands for lengthy and well-understood command sequences, efforts were made to develop software that would automatically schedule the commands given certain input data. By taking the knowledge for *how* commands were to be scheduled and writing algorithms to replicate that knowledge, it was possible to dramatically reduce the number of people and work-hours required to develop a sequence.

Strategies for implementing these automated sequence generators varied from project to project, but the overall concept remained the same. However, previous efforts generally utilized software that was not directly compatible with pre-existing software tools that converted these text-based and human-readable sequences into the binary products that were sent to the spacecraft; or the scheduling approach was clumsy in its implementation. The development of the "autogen" process was the next logical step and was developed on the shoulders of these earlier and sometimes-highly successful efforts. It combines robust scheduling algorithms with software that is compatible with the pre-existing "uplink" software. It literally reduced the duration of some sequence generation processes from weeks to minutes.

What makes it work?

The "autogen" process, as used for Mars Odyssey, included only two major components: APGEN and the "autogen" script.

First, APGEN, or Activity Plan Generator, is a software program that can be used for sequence and mission planning purposes. It is part of the suite of tools developed by the Deep Space Mission System (DSMS) at JPL and is a multi-mission tool, which means that it can be tailored to many different space missions without changing its "core" software. APGEN was designed with a graphical user interface that facilitates the scheduling of activities on a timeline and includes the ability to automatically expand, decompose, and schedule activities. Because it is a multi-mission tool, it is required that an "adapter" write algorithms in the tool-specific language that APGEN utilizes to represent scheduled activities or commands, and how system resources or states are adjusted as a function of time and usage.

Second, the "autogen" script is a mechanism that maximizes the JPL mission operations network environment. It performs very simple tasks, as follows:

- Gathers needed data files from data repositories on the mission operations network.
- Builds other needed data files for the APGEN scheduling algorithms based on inputs specified by the user.
- Sets up the environment to run APGEN, including scheduling instructions.

- Runs APGEN, which schedules the activities and writes the sequences to files.
- Manipulates the resultant files, if needed.
- Initiates any automated sequence processors to prepare the sequence for uplink, if appropriate.

Refer to Figure 1 for a flow chart describing this process.

In essence, the "autogen" script simply sets up the environment for the APGEN software to perform the real work of building the sequence(s). Therefore, the "guts" of the "autogen" process is in the development of the APGEN "adaptation," where all the rules for when and how commands are to be scheduled must be encoded.

It should be recognized that there are several other software tools that APGEN utilizes to perform various functions. Similarly, the "autogen" script calls other scripts to perform its functions. For instance, APGEN relies upon another multi-mission software program developed by DSMS, called "seq_review", to reformat data files into a format that APGEN can understand; and "autogen" relies upon several other scripts to retrieve data files from various locations on the JPL operations network.
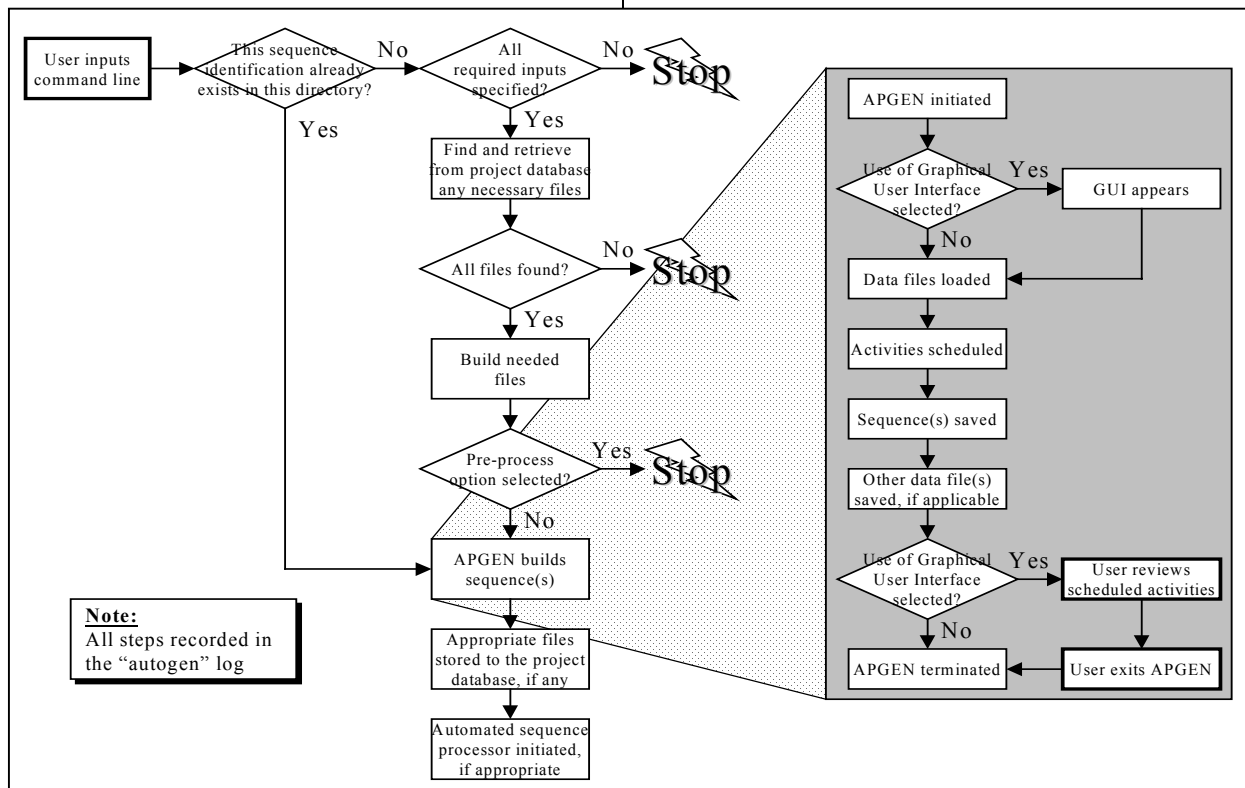

Figure 1: "autogen" Flow Chart.

How was "autogen" used for Mars Odyssey?

The "autogen" process has been used in a variety of different ways for the Mars 2001 Odyssey spacecraft. It has been used during it's cruise, aerobraking, and mapping phases; and in a more limited fashion to build sequences to support the Multiple Spacecraft Per Aperture (MSPA) activities and to build sequences to support the transition period between the aerobraking and the mapping phases. The types of activities that have been scheduled during each of these phases or activities will be discussed, along with a description of some of the challenges that were encountered at each step.

Interplanetary Cruise

The "cruise" phase of a mission is generally defined as the period shortly after a spacecraft is launched from the Earth until before it reaches it's primary destination. These periods are generally quiescent, with short periods of intense activity that may include trajectory correction maneuvers (TCMs) and spacecraft and payload tests or calibrations.

The Mars Odyssey approach to sequencing this phase was to generate a "background" sequence that included the regular commanding that would occur during some pre-defined, long period of time (28 days, in this case). This "background" sequence then provided the foundation upon which the other spacecraft activities were overlaid. In general, before a "background" sequence was designed, an effort was made to plan when the other, more intense activities were to occur in order to ensure that the "background" sequence was compatible.

This approach to sequencing worked rather well, and the "autogen" process readily supported it. During this phase, the "autogen" scheduling algorithms scheduled the following types of commands:
- Calls to a block that commanded the spacecraft to communicate with Earth during appropriate times.
- Commands to incrementally move the solar array to better track the Sun.
- Commands to perform daily, weekly, and monthly flight software diagnostics.
- Commands to perform daily star camera diagnostics.
- Ground directives to cause the sequence modeling software to generate a file that represented the conditions of the spacecraft at a given instant in time, called a "final conditions" file, or FINCON.

The development of the "autogen" algorithms during the cruise phase provided a healthy "learning curve."

For instance, several times it was discovered that the algorithms that were written to describe when the ground antennas were listening to the spacecraft in a "two-way" Doppler mode were very lacking. Three cases arose that forced algorithm updates, as detailed here:
- The times when the Deep Space Network (DSN) antennas are scheduled to listen to a spacecraft are listed in what is called a Station Allocation File (SAF). The stations are generally "allocated" based upon when the spacecraft is visible to the station and depending on other scheduling constraints. However, at each antenna, there is an elevation specified below which the station's transmitter can not be turned on. Therefore, while the station is able to "hear" the spacecraft, it cannot "talk" to it. Roughly speaking, this mode is called "one-way", and it isn't until the transmitter is turned on and the spacecraft receives a signal from the station that the mode is called "two-way." The times for when the stations can "see" the spacecraft and when its transmitter would be above its "transmitter on" elevation limit are specified in what is called a station View Period file (VP). The "autogen" scheduling algorithms originally only took into account the SAF and not the VP file, thereby occasionally commanding the spacecraft to transmit science data during periods that were considered "one-way". This, in itself, is not a problem. Nevertheless, when a transition occurs from the "one-way" to the "two-way" Doppler mode, the station needs to again "lock up" on the spacecraft's signal. During this transition, any data that the spacecraft may be transmitting may be lost. It was desired to avoid sending high-priority data during these transition times, therefore the "autogen" scheduling algorithms had to be re-designed to take into account both data sets (the SAF and the VP file).
- When it is desired to allow a spacecraft to have long periods of time during which it can transmit to the Earth, it is usually necessary to have two or more DSN stations allocated to "track" the spacecraft. As the first station goes out of view of the spacecraft, the second station, located elsewhere, can pick up its signal as the Earth rotates and the spacecraft comes into view. This "station handover" is often transparent to the spacecraft. However, when the "handover" occurs, the same issue, as described above, of needing to be above the transmitter limit, is still in effect, and therefore there are times during these "handovers" when the mode would go from "two-way" to "one-way". As before, it was undesirable to command the spacecraft to transmit high-priority data during the "one-way" mode, so it was necessary to once

again update the scheduling algorithms to command the spacecraft to transmit lower-priority data during those "handovers."

- When the axes of rotation of a DSN station are designed so that it rotates around an axis perpendicular to the surface of the earth (i.e. an azimuth rotation) and then pitches up and down with respect to the horizon (i.e. an elevation axis), the position of the antenna is not always easy to specify. For instance, when the elevation is specified as 90 degrees above the horizon, meaning that the antenna is pointed straight up, the azimuth position is undefined since the antenna can be in any azimuth position and *still* be pointed straight up. Because the antennas can only be articulated at some maximum rate, it is impractical to require a station to track a spacecraft up from the horizon and through the "straight up" position, perform a quick turn-around, and then track it back down the other side. Therefore, the DSN stations that are azimuth/elevation antennas have what is called a "keyhole", or a keep-out zone, which is a maximum elevation above which it won't transmit to a spacecraft. See Figure 2. This keyhole is usually specified to be large enough to allow sufficient time for the antenna to rotate around in order to continue tracking the spacecraft down the second side. This causes another period of "one-way" communication in the middle of what would typically be a continuous "two-way" period. Generally speaking, for a spacecraft to be in such a position that it is possible for a station to track it through the keyhole is very rare, but it did occur to Odyssey during it's cruise phase. Therefore, the "autogen" algorithms had to be updated to once again schedule commands to have the spacecraft transmit lower-priority data during that time frame.

These three examples are outlined here for the intention of showing that there were very specific instances when the scheduling algorithms were insufficient for the purposes at hand and it was necessary to redesign them. Nevertheless, even though
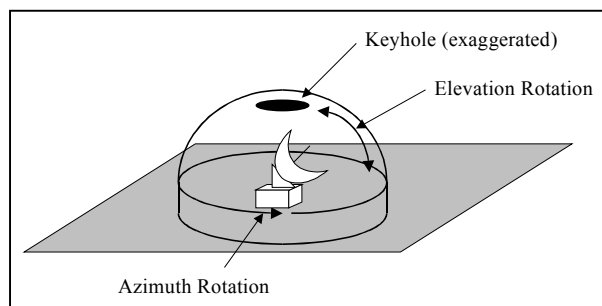

Figure 2: Azimuth/Elevation Diagram with "Keyhole."

the "autogen" scheduling algorithms were a new development, they turned out to be quite robust. In fact, during Mars Odyssey's cruise phase, there were no instances when the background sequence needed to be modified to accommodate the other planned, less regular activities, such as the TCMs or other calibrations. In all cases, these other activities were simply "overlaid" on top of the pre-developed "background" sequences.

During each sequence build, there was a need to send one-time spacecraft commands for general "housekeeping" or spacecraft configuration maintenance purposes. In these cases, rather than modify the scheduling algorithms for the "background" sequence, a short sequence was developed and then "merged" in. In the rare cases when it was actually necessary to modify the "background" sequence, the changes were generally very minor and were made to accommodate special circumstances, and almost never to correct some significant shortcoming of the scheduling algorithms.

There was only one instance during the cruise phase when the spacecraft went into a "safe mode", thereby stopping all of the onboard sequences. Once the situation was resolved, the original "background" sequence was edited to simply remove all the commands before the new desired sequence restart time; and the sequence was again transmitted to the spacecraft.

Aerobraking

The "aerobraking" phase of the Mars Odyssey mission was defined as the period immediately after the spacecraft performed its Mars Orbital Insertion until the spacecraft was placed into its operational orbit. During this period, the spacecraft used the atmosphere of Mars to dissipate its orbital energy, thereby reducing its orbital period "for free" and saving fuel. This phase was characterized by highly unpredictable changes in the spacecraft's orbital period, which was a function of the variability in the density of the Martian atmosphere. With each orbit, the atmosphere would impart a change in the spacecraft's velocity. This "delta-V" was somewhat erratic and therefore made it difficult to predict what the orbital characteristics of the spacecraft would be very far in advance.

At the beginning of the aerobraking phase, the Mars Odyssey navigation team was only able to predict a few orbits ahead to the accuracy desired because the orbit was so large; the spacecraft at this time had an orbital period of approximately 18 hours. During these large orbits, a sequence would be built with a span of

only one "primary" orbit and one additional "backup" orbit. The "backup" orbit concept was implemented to allow the spacecraft to continue aerobraking in the event that the spacecraft operators were unable to transmit the next sequence to the spacecraft for any reason.

As the orbit contracted throughout the aerobraking phase, it was necessary for the navigation team to predict more and more orbits into the future in order to have enough time to prepare a sequence. In fact, at the end of the aerobraking phase, as the orbital period approached two hours, the navigation team was predicting more than 30 orbits into the future. The very first orbit in each of these predictions was a reconstruction of the most recent orbit for which there was reliable data that described the spacecraft's orbital characteristics. Immediately following were several orbits that were extrapolated but not used in the development of the sequence because they had "already occurred" but for which there had been insufficient time to reconstruct, whether for lack of data or due to schedule constraints. Following these were several orbits, also extrapolated, from which the sequences were built. At the end of the aerobraking phase, sequences were being built for six "primary" orbits and three additional "backup" orbits.

The inability to predict the spacecraft's orbital parameters far into the future required that sequences be built and transmitted to the spacecraft very often, sometimes as often as every six hours and sometimes with as little sequence development time as three and one-half hours. Because of this highly constrained schedule, software called the "Automated Sequence Processor," or ASP, was used to automatically process the sequence. This processing included the generation of sequence review files, the construction of the uplink products, and the distribution of all of the appropriate files to the correct locations in preparation for transmitting the sequence to the spacecraft.

To effectively utilize the ASP, the sequencing approach needed to be very straightforward. Therefore, the aerobraking sequences were designed with a heavy dependence on onboard blocks. For each "drag pass", or period during which the spacecraft would pass through its periapsis, there was only one command that needed to be sent to the spacecraft. This command was actually a "block call" that initiated one or more of these onboard blocks. The block call had many parameters that controlled how it should issue commands to the spacecraft, including durations between commands, switches to enable or disable different parts of the block, and filenames that were utilized or otherwise managed by the block. There were only three major blocks that needed to be scheduled during the aerobraking phase of the mission, as follows:
1) the "aero" block, which was the block utilized during each "drag pass",
2) the "abm" block, which was utilized only when an aerobraking maneuver, or ABM, was needed to raise or lower the spacecraft's periapsis, and
3) the "payload_cal" block, which was only used twice during the aerobraking phase to utilize one of the onboard instruments to take an image of Mars.

The need to facilitate quick sequence builds was only part of the problem of sequencing during the aerobraking phase. As mentioned before, with each predicted orbit in the future there was an associated increase in the uncertainty of that orbit's timing. The spacecraft had an onboard software capability to autonomously determine the time of its periapsis passage. Using this "periapsis timing estimator", or PTE, the spacecraft was then able to compare this time with the *expected* time of the periapsis passage and thus calculate the cumulative timing offset from orbit to orbit. It was unable to predict into the future, but it could generally correct for past deviations.

It was desired to utilize this PTE capability to "shift" the sequences around to better center the spacecraft's activities at periapsis (or apoapsis, in the case of the ABMs). However, the spacecraft's command software requires that each command have a time specified at which it should be initiated. This made it impossible to directly have a flexible start time for these block calls. To overcome this problem, the concept of the "seq" blocks was introduced.

The "seq" concept utilized a "parent" block that would be called in the place of the blocks mentioned above. These "seq" blocks had three primary functions:
1) to identify the timing offset of the orbit as calculated by the PTE flight software object,
2) to calculate the time at which the previously mentioned blocks should be kicked off as a function of this offset, and
3) to initiate the blocks at the appropriate times.

The navigation team defined the maximum cumulative uncertainty in the timing of the orbits throughout a sequence, and the "seq" blocks were scheduled this duration earlier than the "regular" blocks would have been scheduled. To illustrate this capability, see Figure 3 and Table 1.
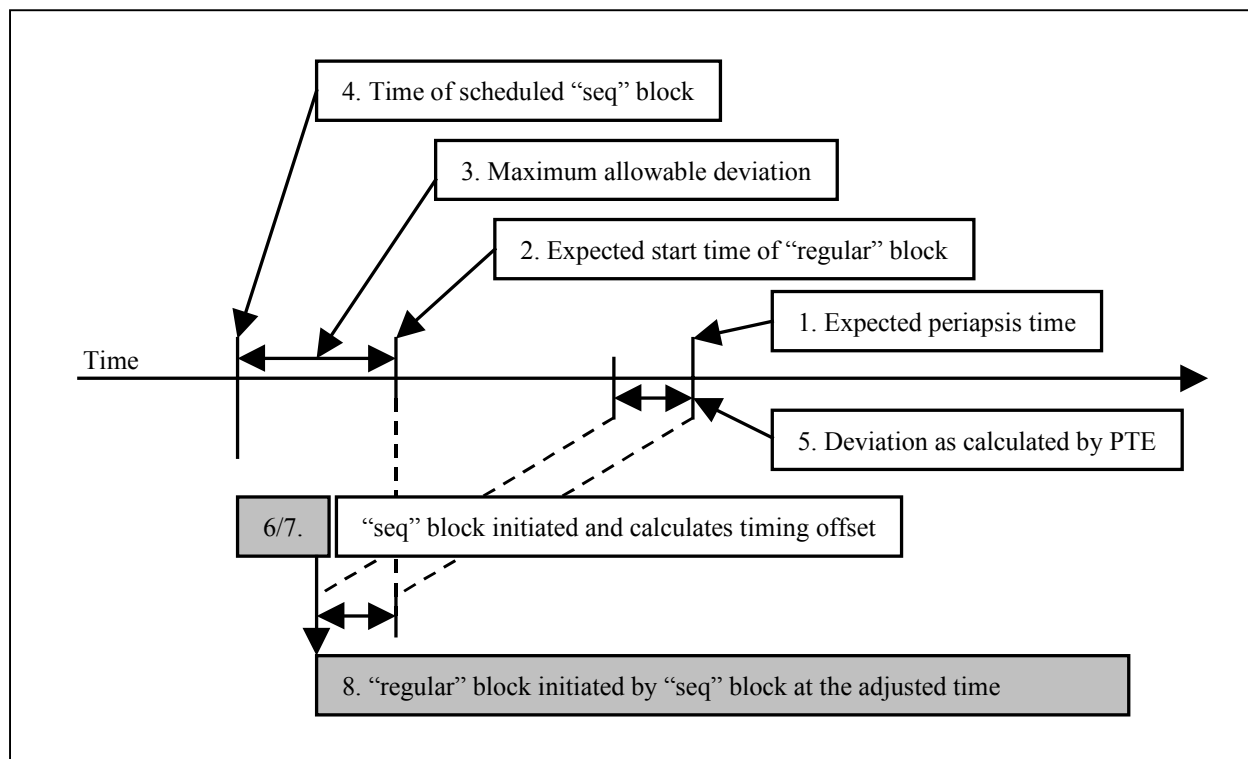
Figure 3: How The "Seq" Concept Is Utilized to Shift a Block Call.

Table 1: How the "Seq" Concept Works

1.	Navigation team estimates expected time of periapsis passage.
2.	Expected start time of "regular" block call is calculated.
3.	Maximum allowable deviation for the sequence is determined.
4.	"seq" block is scheduled at expected start time of "regular" block minus this maximum deviation.
	Sequence is sent to the spacecraft.
5.	PTE determines experienced timing deviation.
6.	"seq" block is initiated onboard the spacecraft.
7.	"seq" block calculates time to initiate "regular" block based on expected start time minus the actual deviation.
8.	"seq" block initiates the "regular" block at the adjusted time.

The use of this "seq" structure allowed the development of "autogen" scheduling algorithms that were quite robust. In addition, the APGEN software was utilized in a "batch" mode, disabling its graphical-user interface, and coupled with the capabilities of the ASP. This facilitated the rapid development of sequences during the aerobraking phase; with the issuance of one command line, a sequence was automatically built, modeled, converted to the uplink binary file, and distributed to the DSN in preparation for uplink in less than twenty minutes.

The "autogen" tools scheduled the following types of commands:

- The "seq" blocks, in three varieties: the "aero_seq" block for the "drag pass", the "abm_seq" block for the ABMs, and the "themis_seq" block for the science observations.
- A command to "kill" any previous sequence that may be operating.
- Ground directives to suppress the transmission of commands to the spacecraft during times when the antenna was not pointed towards Earth, as calculated from the scheduled blocks.
- Ground directives to cause the sequence modeling software to generate FINCONs.

The "autogen"/ASP combination turned out to be remarkably robust and there were only three command

builds (in over 200 builds!) throughout the entire aerobraking phase during which the "autogen" tools were called into question. In the end, each of these instances turned out to further validate the tool and concept and showed that it behaved exactly as it should have. The following outlines these instances:

- As shown before, the "autogen" scripts were designed to perform a variety of functions before it attempted to build a sequence, namely collecting and building some data files. If a user attempted to perform a "re-build" of a sequence in a subdirectory where some of the data files already existed, the "autogen" scripts were designed to not attempt to re-collect and re-generate the files, but to simply run the scheduling algorithms on the pre-existing files. This was an intentional design feature that allows a user to perform sequence re-builds very easily if only one or a few of the many input files have changed from previous sequence builds. During aerobraking, each sequence was built with one or more "primary" orbits and one or more "backup" orbits. Because the "backup" orbits were contingency orbits, when a new sequence was sent to the spacecraft the first thing it would do was stop the pre-existing sequence before it had the chance to initiate any of the commands for the "backup" orbits. In order to do this in a timely fashion, it was necessary to track when each of the block calls for the "aero_seq" blocks were scheduled. In addition, the "killing" of the pre-existing sequence was always intended to be scheduled at least a minute before the next "aero_seq" block call would occur. In cases where a sequence was re-generated, the "backing up" of the "kill" would increment earlier and earlier, which was an intended behavior. During one particular build, the user noticed this effect and became alarmed since they did not understand the behavior of the timing shifts. Therefore, in order to prevent this effect from occurring again, the sequence build procedure was modified to ensure that whenever a sequence needed to be re-generated, the user would start "from scratch" with an incremented sequence identification name, thereby avoiding the issue altogether. To the user, this issue initially appeared to be a problem with the "autogen" scheduling algorithms because the commands were being scheduled earlier than expected, but eventually it was concluded that the issue was a weakness in the sequence generation process.

- In order to provide flexibility to the "autogen" scheduling algorithms, many of the commands that are scheduled had parameters specified that would change *how*, *when*, or *if* they were scheduled. These parameters were read into APGEN, which then interpreted those parameters and scheduled the commands appropriately. During aerobraking, it was desired to re-evaluate on a weekly basis how "autogen" scheduled the block calls. Therefore, the concept of the "reset sheet" was implemented to allow a specific set of parameters to govern how the sequences were developed over the course of a week. Many of the parameters on this "reset sheet" were parameters that were simply passed to the blocks themselves. Other parameters were specific filenames that needed to be sent to the spacecraft for command purposes. In some cases, it was desired to allow several filenames to be listed for a particular parameter and then to have the "autogen" schedulers extract the correct filename based upon the orbit number and/or some other parameters. In this way, it was possible to specify many filenames for a single parameter that could be used over the course of a week without having to revise the "reset sheet". Constructing arrays that would facilitate this multiple parameter capability was designed as part of the "reset sheet". In one instance, one of these arrays in the reset sheet had been incorrectly specified and the sequence build failed. The "autogen" algorithms had been designed to terminate the sequence build process in just such an eventuality, and it took some investigation to determine the root cause of the termination. The solution was simply to correct the array in the reset sheet. Once again, the scheduling algorithms were called into question, but it turned out that "autogen" was behaving correctly.

- As aerobraking progressed and the spacecraft's orbital period became shorter and shorter, occasionally it was discovered that the sequences from one orbit to the next nearly "overlapped" each other. To prevent against this eventuality, much up-front analysis was performed to ensure that the "reset sheet" was correctly specified and would be valid throughout the entire "reset period," usually one week in duration. However, some of this analysis was done without taking into account the uncertainty in the timing of the orbits. The "autogen" scheduling algorithms would output a ground directive to write a FINCON at the expected end of the block call for each "drag" pass *plus* the total accumulated uncertainty of the orbital timing, as discussed above. For subsequent sequences, the times of these FINCONs were used to determine the start time of the sequences. For the shorter orbits when the sequences nearly overlapped each other, these FINCON directives were actually scheduled to occur *after* the scheduled time for the next "aero_seq" block call. This condition was actually acceptable for a given

sequence. The problem lies in the development of the next sequence. When it came time to build the next sequence, the start time would be extracted from the previous FINCON, as expected. However, due to the "backing up" of the commands to "kill" the previous sequence in a timely fashion, as mentioned earlier, the first commands in the new sequence would not be output to the sequence file because their start times would be prior to the sequence start time. This condition was unacceptable, and it was decided to adjust how the FINCON directives were scheduled by "autogen" in order to remove the delay in their scheduled times based upon the accumulated uncertainty of the orbital timing. This solved the problem for most of the final sequences, but there were a few instances when the times of the FINCON directives had to be manually moved by the sequence engineer to avoid this problem. For this issue, even though a change to the scheduling algorithms was required, it was not related to how the spacecraft commands were scheduled, only in the FINCON ground directive, which facilitates ground modeling.

The remarkable performance of the "autogen" strategy is even more impressive considering that this was a "first use" situation; the sequence build process had never previously been directly tied to the ASP. In addition, it is noteworthy that the "autogen" strategy was readily capable of supporting the highly complex nature of the sequence structure during this mission phase. With very little early testing and on a short turn-around basis, the "autogen" algorithms were developed, deployed, and fully utilized to support this major mission phase in a profoundly successful manner.

Mapping

The "mapping" phase of the Mars Odyssey mission was defined as the period after the aerobraking phase had completed and once the spacecraft had achieved its operational orbit and until the end of the prime mission. This phase tends to be very regular in its activities, but also very active. It is characterized by the same types of "housekeeping" activities as during the cruise phase, with the added complexity of orbital geometric events, such as Earth occultations and Solar eclipses, and with the demands of facilitating the mission's science objectives.

The Mars Odyssey approach to sequencing this phase is very similar to the sequencing approach used during the cruise phase, and once again includes the use of a "background" sequence that performs the regular,

engineering commanding of the mission for a long period of time (once again, 28 days). In addition, science activities are overlaid on this sequence and there are occasional periods when other engineering activities must be scheduled.

The "autogen" approach works as well for the mapping phase as it did for the cruise phase. The scheduling algorithms schedule the following types of commands:

- Calls to a block that commands the spacecraft to communicate with Earth during appropriate times.
- Commands to perform daily, weekly, and monthly flight software diagnostics.
- Commands to perform daily star camera diagnostics.
- Calls to a block that cause the reaction wheels to be desaturated.
- Commands to change the rate at which data is transmitted to the Earth based upon the diameter of the DSN antenna that is allocated.
- Ground directives to suppress the transmission of commands to the spacecraft during times when it's antenna is not pointed towards Earth, when the spacecraft's low-gain antenna is selected as the primary receiver, and when Mars occults the spacecraft's view of Earth.
- Ground directives to cause the sequence modeling software to generate FINCONs.

At the time of this writing, the mapping phase has been progressing smoothly. There have been no major problems with either the spacecraft or the "autogen" process or tools.

MSPA and Relay Coordination

During the aerobraking and the mapping phases of the Mars Odyssey mission, there have been times when the Mars Odyssey spacecraft and the Mars Global Surveyor needed to "share" the same DSN stations in order to perform their missions. The technique of having a single antenna "listen" to more than one spacecraft at a time, and transmit to one of them, is called "Multiple Spacecraft Per Aperture", or MSPA.

While modeling a sequence for a spacecraft, a file is generated that contains "keywords" which are used by the DSN operators to ensure that the antenna tracks and transmits to the right spacecraft at the right time. Each spacecraft produces one set of these instructions for a period of time. Since each station is usually only allocated to track one spacecraft at a time, the sets of instructions that are sent to a single station from multiple spacecraft almost never overlap temporally, thereby avoiding any contradicting "keywords".
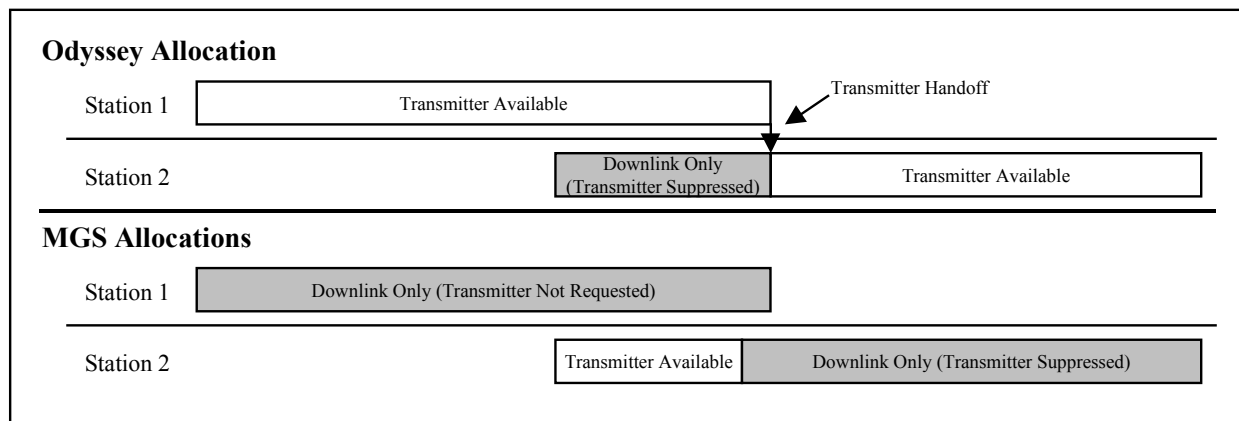
Figure 4: The MSPA Allocation Pattern for Mars Odyssey and Mars Global Surveyor.

However, in the cases of MSPA passes, it is necessary to ensure that a given station is instructed to transmit to only one spacecraft at a time, even though the station is capable of receiving data from multiple spacecraft simultaneously. In order to do this, a ground directive must be produced for the additional spacecraft that "suppresses" the use of the station transmitter during that period. This ground directive is translated into the "keywords" that are provided to the DSN station, thereby ensuring that the antenna is instructed to transmit to only one spacecraft at a time.

To appropriately schedule these "suppressions", heavy coordination must take place between the operations teams for the multiple spacecraft. Some decisions must be made as to how long the transmitter will be allocated to each spacecraft during the DSN pass. During the Mars Odyssey aerobraking phase, there was a regular pattern to how the station's transmitter time was allocated to each spacecraft. Refer to Figure 4 for a diagram outlining this pattern.

As can be seen in the diagram, both stations were allocated to both spacecraft during the same period of time, with an overlap of several hours between station allocations. For Station 1, the station was allocated to Mars Odyssey with a request for the transmitter, while for MGS it was allocated with *no* request for the transmitter. For Station 2, the station was allocated with the transmitter requested for *both* spacecraft.

Under normal circumstances, the "keyword" file that was generated for Mars Odyssey would have instructed the transmitter for Station 2 to begin transmitting to the spacecraft at the very beginning of its allocation. The same would have happened for MGS because its allocation for Station 2 also requested the transmitter.

This condition would have caused contradictory "keywords" to be sent to the DSN station, erroneously instructing it to transmit to both spacecraft. Because there was a requirement to have continuous transmitter coverage for Mars Odyssey during its aerobraking phase, it would have been necessary to suppress Station 2's transmitter for MGS throughout its allocation, thereby denying MGS any transmitter time at all in this scenario.

The approach illustrated in Figure 4 allowed MGS to have at least some transmitter time where previously there would have been none. This was accomplished by utilizing the tail end of Station 1's allocation for Mars Odyssey, which would have previously gone unused, by suppressing Station 2's transmitter during the beginning portion of its allocation to Mars Odyssey, thereby delaying the "transmitter handover" until the end of Station 1's allocation. This "down time" for Station 2's transmitter was then free to be utilized by MGS. To ensure no overlap in the transmitter times for each spacecraft, the latter half of Station 2's allocation to MGS needed to be suppressed early enough to turn off the transmitter and allow time for the station to be reconfigured in time to transmit to Odyssey.

It was originally expected that the use of this MSPA capability would occur quite often. Therefore, an "autogen" process was developed to perform the following tasks:
1) Read in the SAF and VP files for both spacecraft.
2) Determine if the transmitter had been requested for each allocation.
3) Determine if the same transmitter had been requested for both spacecraft.

4) Schedule a transmitter suppression for the beginning of the overlapping transmitter request until such time as the "outgoing" allocation concluded for Mars Odyssey.
5) Schedule a transmitter suppression from some period of time before the end of the previous suppression until the end of the overlapping transmitter request for MGS.
6) Construct a distinct sequence file for each spacecraft containing the DSN transmitter suppression directives.

In practice, it was only necessary to perform MSPAs three times throughout Odyssey's aerobraking phase and it would have been very simple to construct these DSN transmitter suppression directives manually. However, the development of the MSPA capability provided two major benefits, as follows:
1) The ability to read in data sets for multiple spacecraft was a new capability. Previously, the "autogen" algorithms made no distinction between SAFs for multiple spacecraft, and would simply assume that all of the data applied to one spacecraft.
2) The ability to write out multiple sequence files was also a new capability. Previously, the "autogen" algorithms would only cause APGEN to write out one sequence file for a pre-specified spacecraft.

Both of these new capabilities will be extensively used in the near future to perform "relay" coordination between Mars Odyssey and the twin Mars Exploration Rovers, which will land on Mars near the beginning of 2004. The rovers will transmit data to Odyssey, which will store the data for a short time and re-play it back to Earth at its earliest opportunity. These two capabilities will allow scheduling algorithms to be developed to automatically schedule the commands and activities that must occur on all three spacecraft to make this happen. In addition, it is expected that not only will these algorithms have to coordinate activities between these three spacecraft, but that coordination must also occur between other orbiting and landed spacecraft on Mars during that timeframe, such as MGS, Mars Express Orbiter, and Beagle II. It is very clear that the development of these abilities to manage data sets for many spacecraft will provide long-ranging benefits in the future when there will be increased coordination between multiple projects and greater contention for ground resources, such as DSN allocation time.

Transition from Aerobraking to Mapping

Immediately following Mars Odyssey's aerobraking phase, there was a period of nearly two months called the "transition to mapping" period. During this period, the spacecraft was primed to perform its principal mission by deploying additional hardware and performing calibrations. There were no standard sequences that needed to be operational during this time, and the spacecraft was left in a fairly quiescent state, with the exception of the unique and specially planned activities that needed to occur.

At the beginning of this transition period, the ground stations were allocated in such a way as to be continuously communicating with the spacecraft except when Mars occulted the Earth. In addition, the ground operators were scheduled to be available to operate the spacecraft twenty-four hours a day. During this "continuous coverage" period, it was very easy for the ground operators to interactively instruct the spacecraft to transmit high-priority data when the spacecraft was in "two-way" communication with the Earth, and to avoid transmitting this high-priority data at other times. When the continuous staffing and DSN allocation periods came to an end, the spacecraft would be left in a state where it would attempt to transmit the higher-priority data regardless of whether there is a ground station visible and allocated to receive the data. In this way, some high-priority data could have been lost.

This condition was recognized near the end of the aerobraking phase, and it became desirable to put a very simple "background" sequence onboard the spacecraft that would instruct the spacecraft to transmit the high-priority data only when appropriate. Therefore, an "autogen" process was rapidly deployed to schedule these changes. By adapting the scheduling algorithms from the previously mentioned phases, it was relatively straightforward to develop an automated process to build these "background" sequences.

The resulting sequences were very simple and literally only contained commands to change the type of data that the spacecraft would transmit. Nevertheless, it was discovered that the "autogen" modeling did not schedule the commands correctly in every case when there were station "handovers." However, it was only about 2% of the scheduled commands that had to be manually adjusted before the sequence was ready to be transmitted to the spacecraft.

Even though the "autogen"-developed sequence wasn't perfect, it still dramatically reduced the amount of effort that would have been required to build these sequences manually. Therefore, this effort admirably illustrated how the "autogen" process was rapidly adapted to solve a potentially daunting command scheduling problem that would otherwise have required much manual effort.

What lessons were learned?

The development of the "autogen" process for Mars Odyssey has provided an excellent case study to illustrate that properly developed scheduling algorithms can dramatically reduce sequence development time and effort. In addition, the foundation provided by the Mars Odyssey adaptations readily serves as the foundation for sequence development efforts for other spacecraft. For example, a brief demonstration for the Genesis spacecraft, also managed by the Jet Propulsion Laboratory, showed that the "autogen" process could be quickly adapted and fully functional for another mission in as little as 30 work-hours.

It is essential to clarify some of the considerations taken when developing the "autogen" process for Mars Odyssey that make this kind of flexibility possible, as follows:

- Keep the sequencing approach simple. If it was discovered that a series of commands was being repeated consistently, then the appropriate development of a block was a powerful tool in simplifying sequences and reducing the size of command loads. In addition, blocks, when designed and utilized intelligently, dramatically streamlined the scheduling of commands within sequences. Nevertheless, the proper balance had to be found between the complexity of the available blocks and intelligent sequencing.

- Parameterize everything. Every aspect of the scheduling algorithms that could have been parameterized was parameterized and made accessible to the user. In this way, the user retained control over *how* things were scheduled without being required to directly adjust the scheduling algorithms when small changes were needed. In practice, it was useful to encode several techniques for scheduling the same commands to provide additional flexibility. For instance, during the mapping phase for Mars Odyssey, the user had the ability to schedule DSN communications commands on a once-per-orbit basis, or schedule them whenever a DSN station is in view. In addition, the range of parameters for Mars Odyssey also included, among other things, the durations between certain commands and switches to schedule or not to schedule an activity. Having all these "knobs and dials" available to the user provided great flexibility when developing a sequence.

- Do your work up-front to analyze the scenarios. If the sequencing approach to a mission phase is analyzed thoroughly enough, it is conceivable that a sequence scheduler could be developed prior to

execution and never be modified again. However, in practice, many circumstances may require modifications to the sequence scheduling software. Nevertheless, early and significant efforts to detail how a mission phase should be sequenced may dramatically simplify the efforts required to actually build a sequence and reduce the amount of last-minute coding that is often required.

- The scheduling algorithms rarely produce a perfect sequence, particularly for long sequences. The more complex the sequence, the more likely that errors exist. If it discovered that there are minor errors with the sequence, there should be no reason why the automatically scheduled sequence shouldn't be manually edited to fix it. If the scheduler is systematically not scheduling the sequences correctly, then there should be no fear of updating the scheduling algorithms in order to avoid being required to make many manual edits to the sequence.

- The "autogen" approach has been designed exclusively to build sequences; it has not been designed to supplant any sequence checking efforts that must be performed on a sequence prior to being uplinked to a spacecraft. It has been discovered that when developing the scheduling algorithms for Mars Odyssey, only those models that are necessary for the scheduling of the commands are required to be accessible by the scheduling algorithms. It has proved advantageous to keep the functions of sequence generation and sequence checking separate and distinct. The function of the "autogen" schedulers is to build a sequence that meets the *intentions* of the sequence developers, whereas the function of any sequence checking software is to ensure that the sequence is safe to be transmitted to the spacecraft. Linking the two functions could complicate configuration management efforts, compromise spacecraft safety, and reduce the flexibility needed to occasionally change the scheduling algorithms.

- Don't place the sequence schedulers under configuration management until things have stabilized. For Mars Odyssey, it was discovered that even early efforts to develop the sequence scheduling software were unable to anticipate some of the realities of actually operating the spacecraft. For instance, during Mars Odyssey's mapping phase, it was quickly discovered that the sheer size of a 28-day sequence was simply too large for the spacecraft's onboard sequence management software to handle. A short-term solution required that the 28-day sequence be cut into four pieces, each 7 days in duration. The long-term solution was the development of two

new spacecraft blocks that would dramatically reduce the number of commands that needed to be sent to the spacecraft to achieve the same effect, and the reduction of the length of many of the onboard block names. Had the schedulers been placed under configuration management early, it would have been more difficult to adjust them in a timely fashion to meet the changing needs of the spacecraft operators.

- The algorithms will do exactly what you tell them to do. Indeed, even though the "autogen" scheduling software has worked dramatically well, all instances where this process has failed has been the result of bad input files, bad input parameters, or algorithms that were not designed properly in the first place.

Can you put this automated sequencing onboard the spacecraft?

It is probably better to ask, "*Should* you put this automated sequencing onboard the spacecraft?" This document is not intended to be a discussion on the benefits of ground automation versus onboard autonomy; nevertheless, some discussion is warranted.

Historically, the development of high-level spacecraft autonomy to control a spacecraft has proven to be expensive and questionably useful. However, onboard autonomy has been used in the past to great success to perform certain low-level activities, such as attitude and orbit control, power and thermal management, and payload monitoring. Nevertheless, onboard autonomy often has the following disadvantages:

- Onboard autonomy is often programmed as part of the flight software. Therefore, when updates to the algorithms are needed, a flight software "patch" or complete code replacement is necessary. These changes are generally high-risk activities.
- Comprehensive autonomy that will be fully capable of handling all or even most circumstances and eventualities is difficult to design. This makes the development of high-level autonomy prohibitively time-consuming and expensive. In addition, as the autonomy becomes more and more capable (i.e.: complex), the testing of that same autonomy becomes exponentially more difficult. Thorough testing of these algorithms is often time-consuming and expensive.
- It is very difficult to model on the ground *a priori* the behavior of onboard autonomy since, by its very nature, it utilizes its real-time environment to determine its next course of action. This translates directly into difficulty in generating predictions about the behavior of the spacecraft for users on the ground. In cases where ground interaction is

required, such as with regards to coordinating ground events such as DSN passes, the identification of seemingly arbitrary science observation targets, etc.; this can be a major problem.

Instead of utilizing onboard autonomy, the use of ground automation can often overcome many of these issues, as follows:

- Since ground automation can be developed using non-compiled and scripted algorithms, it is more accessible than onboard flight software, thereby allowing changes to be made more readily. In addition, the act of merely updating the algorithms does not put the space vehicle at risk.
- Ground automation is generally less comprehensive than onboard autonomy is often required to be, being programmed to handle *most* scenarios, rather than being encoded to handle *all* pre-conceived scenarios. In cases where the ground automation may be lacking, human intervention is still available to troubleshoot problems. Because the range of behaviors of ground automation is generally more limited, testing the algorithms becomes simpler than testing a similar onboard autonomy and can be independent of flight software testing.
- By generating command sequences using ground automation, sequences become largely deterministic, thereby making ground modeling more capable of predicting the spacecraft's behavior. As the sequences must be developed prior to transmission to the spacecraft, there is room for human intervention to add to or modify the sequence.

Each strategy for controlling the spacecraft has its place. Onboard autonomy for controlling a spacecraft can be very useful for closed loop, bounded activities that are highly dependent on the spacecraft's environment and don't require ground interaction or modeling. Ground automation of command sequences are very useful when ground interaction is required, modeling is a necessity, and when the spacecraft's interaction with its environment is well understood and relatively predictable. In addition, ground automation generally has a much lower cost for development and maintenance and more readily provides for ground-in-the-loop activities. A good spacecraft design will incorporate both command strategies in a well-balanced, intelligent manner to improve spacecraft operability, reduce development *and* operating costs, and reduce mission risk.

So is "autogen" useful?

Since it was first deployed in early 2001, the "autogen" process and tools have built 7 cruise sequences, over 200 aerobraking sequences, 6 transition-to-mapping sequences, and, to this writing, 13 mapping sequences (and counting). In all this time, there have been no major failures of the tool or process, but there has been a dramatic reduction in the effort that has been required to build these sequences. In years past, sequence system engineers spent many days calculating when commands should be scheduled and manually constructing these sequences. With the use of the "autogen" process, these efforts are largely relegated to the past, with the time of the sequence system engineers being better spent verifying the intent and safety of the sequence. It is anticipated that future missions will utilize the "autogen" process or its descendents to continue to automate the sequence development process, thus enabling safer, more cost effective missions.

Brief Acronym List and Glossary

ABM	Aerobraking Maneuver
APGEN	Activity Plan Generator
ASP	Automated Sequence Processor
Autogen	Automatic sequence generation process and/or script
Block	A sequence that may be reusable and may include input parameters to change the way it issues commands; similar to subroutines.
DSMS	Deep Space Mission System
DSN	Deep Space Network
FINCON	Final Conditions (file)
GUI	Graphical User Interface
JPL	Jet Propulsion Laboratory
MGS	Mars Global Surveyor
MSPA	Multiple Spacecraft per Aperture
PTE	Periapsis Timing Estimator
SAF	Station Allocation File
Sequences	Files that contain a series of commands, each with a specified time, that will be sent to the spacecraft.
TCM	Trajectory Correction Maneuver
VP	View Period (file)